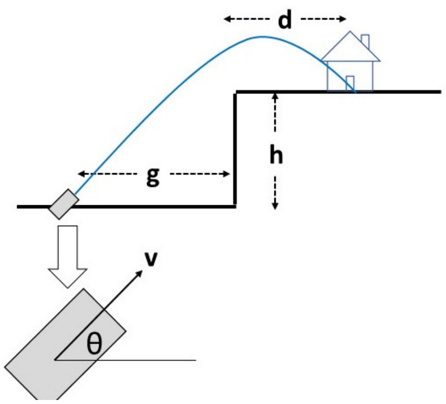
Project #1, Monte Carlo Simulation

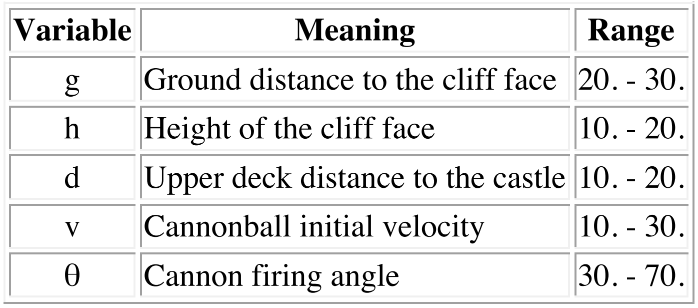
Vincent Rastello
rastellv@oregonstate.edu

Project 1 Summary:

For this Project we are running a Monte Carlo Simulation on a scenario. A Monte Carlo Simulation is used to find the range of outcomes for a series of parameters. Our scenario is a band of mercenaries are attempting to destroy a castle on top of the cliff. They can only determine the 5 input parameters within certain ranges. I have parallelized the given algorithm attached in the report. Using that algorithm I ran the parameters through a series of different trial amounts with different amounts of threads used. The probability of the mercenaries destroying the castle was determined for each trial amount and thread count. The peak performance was also calculated per run using megatrials per second (millions of trials per second).

Ranges for input parameters and drawing of scenario:





Variable	Meaning	Range
g	Ground distance to the cliff face	20. - 30.
h	Height of the cliff face	10. - 20.
d	Upper deck distance to the castle	10. - 20.
v	Cannonball initial velocity	10. - 30.
θ	Cannon firing angle	30. - 70.

Project 1 Results

See the table below for the number of threads vs. number of Monte Carlo Trials, with the MegaTrials/Second values filled in.

Project #1, Monte Carlo Simulation

Vincent Rastello
rastellv@oregonstate.edu

		# of Monte Carlo Trials							
		1	10	100	1000	10000	100000	500000	1000000
	1	0.8	6.45	4.59	4.8	4.8	8.75	8.72	8.85
	2	0.45	2.95	8.47	9.36	9.5	17.47	17.3	17.51
# of threads	4	0.59	4.3	21.23	32.81	19.03	18.78	34.83	34.53
	6	0.43	3.38	19.36	42.6	47.18	28.51	52.13	51.94
	8	0.37	2.81	23.08	49.97	58.73	31.83	61.56	69.34

This simulation was ran with 100 numtries, recording the peak performance for those tries. The results below indicate that as the number of threads went up the performance went up. For a single trial and 10 trails, the performance was negligible, this is because with only a few threads doing the program was not able to take advantage of the parallelism. Once you get up to 100 trials you see an increase in performance going somewhat linearly as the number of threads increases. The curves become more linear as the trail count increases. **The algorithim** worked the way it did because I added parallel threads to do the work of the for loop, I used shared variables that were declared before the loop then reduction variable for the incrementing sum (NUMHITS), this is to prevent collisions for loads of NUMHITS

Based on the run with 8 threads and 1000000 trials the Probability of the Mercenaries hitting the Castle is: 6.54%

*Fp (F Parrallel) Calculation:

*For values I will be using the 1000000 trial run.

n (number of threads) = 8
s (speedup) = (Performance with 8 threads)/(Performance with 1 thread)

$$F_p = \frac{n}{n-1}\left(1 - \frac{1}{\frac{MegaTrialsPerSec_n}{MegaTrialsPerSec_1}}\right)$$

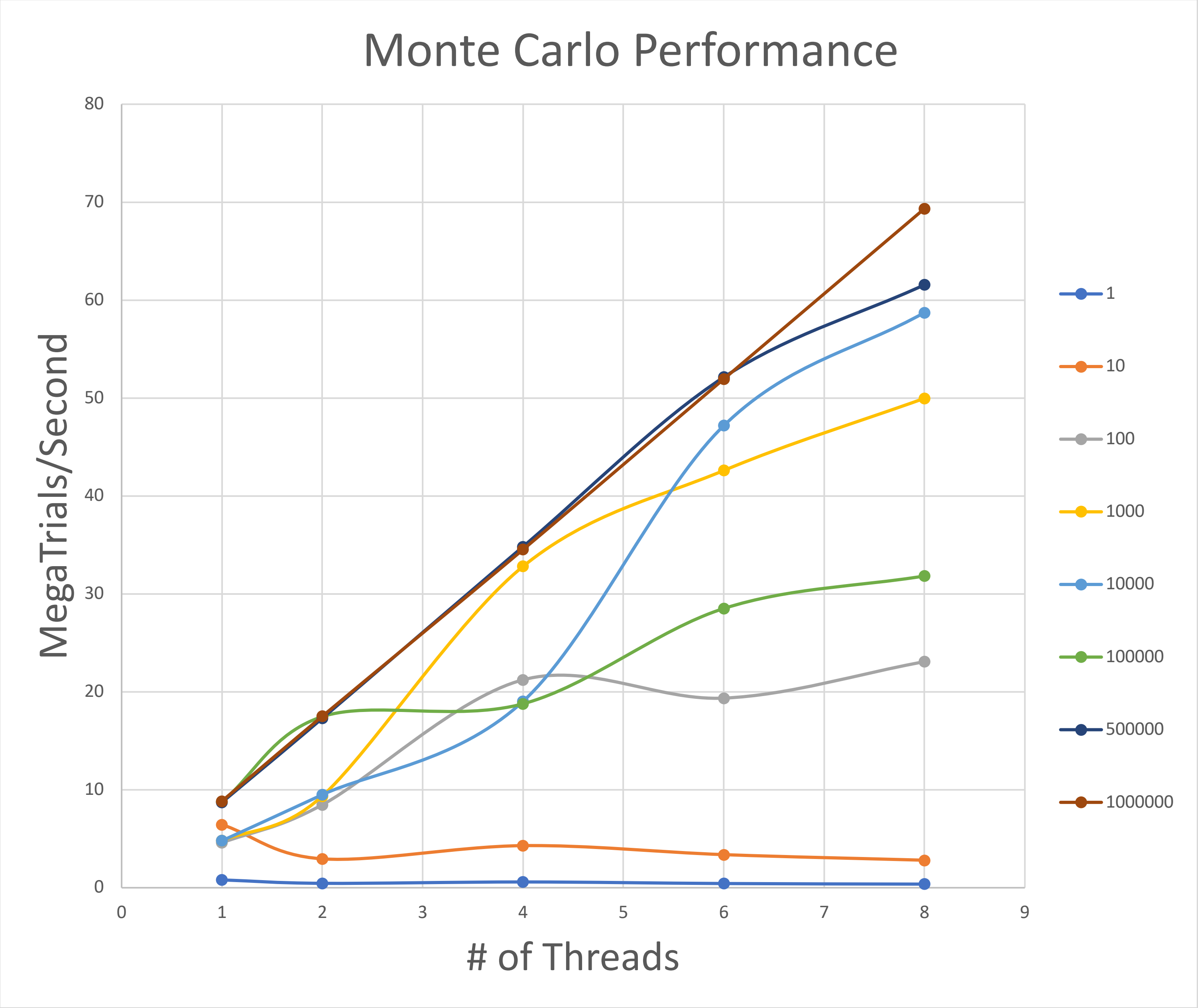
$$= \frac{n}{n-1}\left(1 - \frac{MegaTrialsPerSec_1}{MegaTrialsPerSec_n}\right)$$

$$= \frac{8}{7}\left(1 - \frac{8.85}{69.34}\right)$$

$$F_p = 99.69\%$$

Project #1, Monte Carlo Simulation
Vincent Rastello
rastellv@oregonstate.edu

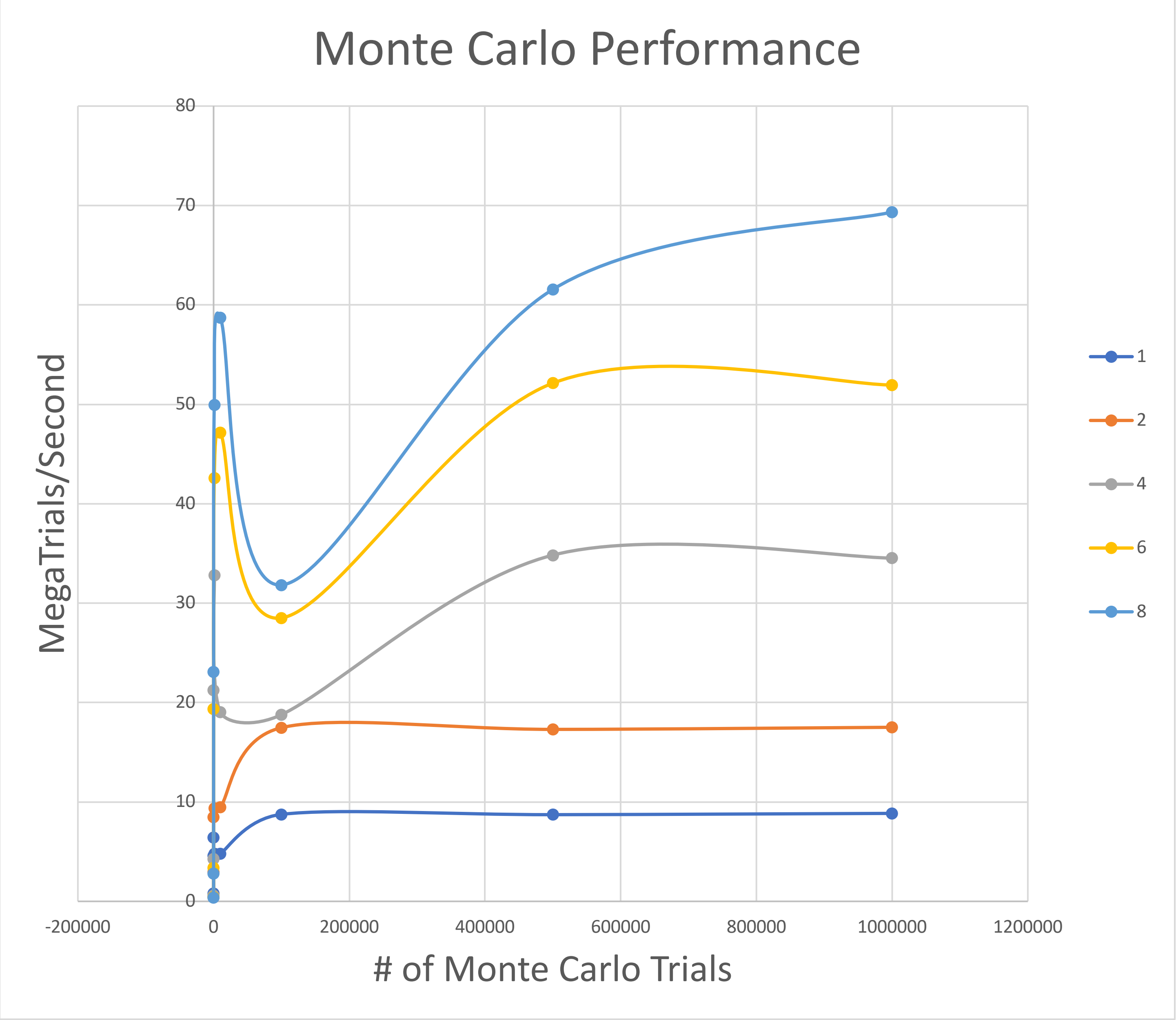
Colored lines are Number of Trials



Colored Lines represent # of threads

Project #1, Monte Carlo Simulation
Vincent Rastello
rastellv@oregonstate.edu

The performance for 6 and 8 threads dipped between 0 and 200000 trials, this may be because I was using the OSU server and the load times were high. (4 4 4). These curves are asymptotic because you get diminishing returns as you increase trial count. You can only really benefit from more threads at that point.


Monte Carlo Performance
MegaTrials/Second
# of Monte Carlo Trials